VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační panel pro Windows 8

Windows 8 Information Panel

2015

Ondřej Boháč

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Boháč**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Informační panel pro Windows 8
Windows 8 Information Panel

Zásady pro vypracování:

Cílem práce je navržení a vytvoření aplikace pro prezentaci multimediálního obsahu v prostředí Modern operačního systému Windows 8.1.

1. Zmapujte a popište možnosti prostředí Modern z pohledu vývoje aplikací
2. Navrhněte aplikaci, která na základě specifikovaných podkladů (XML) bude schopna autonomně prezentovat obsah v čase
3. Implementujte tuto aplikaci s využitím technologií .NET
4. Zhodnoťte výslednou aplikaci a možnosti jejího dalšího rozšiřování

Seznam doporučené odborné literatury:

[1] P. Lubbers, HTML5 - Programujeme moderní webové aplikace, 2011, CPRESS, ISBN: 9788025135396
[2] S. Walther, Windows 8.1 Apps with HTML5 and JavaScript Unleashed, 2014, Sams Publishing, ISBN: 978-0672337116
[3] J. Liberty, Pro Windows 8.1 Development with XAML and C#, 2014, Apress, ISBN: 978-1430240471

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2015 .....

Tímto bych rád poděkoval Ing. Michalu Radeckému Ph.D za odborné vedení práce a také za čas a trpělivost, kterou mi věnoval při konzultacích. Dále bych rád poděkoval rodičům, kteří mi umožnili studium a byli mi podporou. Děkuji všem, kteří mi s prací pomohli jakýmkoliv způsobem.

Abstrakt

Cílem této práce je navržení a vytvoření aplikace pro prezentaci multimediálního obsahu, sloužící jako digitální informační panel pro všeobecné použití. Aplikace tento obsah bude zobrazovat podle scénáře vytvořeného specificky pro dané použití. Koncová aplikace bude vytvořena pro platformu Windows 8 s použitou knihovnou WinJS. V práci se také zaměřím na další technologie jako je TypeScript.

Klíčová slova: Informační panel, WinJS, Windows 8, HTML5, Java script, Universal App, Type script, Less

Abstract

Purpose of this thesis is design and implementation of application, which presents multimedia content. It serves as a digital info-panel for general usage. Content is shown according to created scenario, fitting specific usecase. Application is developed for Windows 8 platform with usage of WinJS library. Further on I focus on other technologies, such as TypeScript.

Keywords: Information panel, WinJS, Windows 8, HTML5, Java script, Universal App, Type script, Less

Seznam použitých zkratek a symbolů

HTML	–	Hyper Text Markup Language
XML	–	Extensible Markup Language
CSS	–	Cascade Style Sheet
RSS	–	Really Simple Syndication
GUI	–	Graphical User Interface
GPS	–	Global Position System
API	–	Application Programming Interface
DOM	–	Data Object Model

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce je navrhnout a vytvořit aplikaci pro prezentaci multimediálního obsahu, který bude prezentován formou informačního panelu. Aplikace bude vyvíjena pro operační systém Windows 8, za použití knihovny WinJs společnosti Microsoft.

Vývoj aplikace není však pouze o psaní kódu, primární je znát očekávání a způsob jakým bude aplikace používána. Dále bude následovat popis plánování stavby celé aplikace, která bude postavena na typových příkladech z reálného světa.

Přecházíme do fáze, kdy návrh aplikace je hotový a vše připraveno pro začátek práce. Následují poslední kapitoly, popisující technické detaily vývoje. Primárním cílem je tedy zaměřit se na logickou část aplikace, řešící navrženou funkčnost. Hotová aplikace bude demonstrována ukázkou příkladů z reálného světa.

Poslední části je závěr. Důležité je zhodnocení aplikace, zda plní stanovené předpoklady a odpovídá vytyčenému návrhu. Nejedná se však jen o hodnocení vývoje a dosažené funkčnosti, ale celé práce a hlavně zda použité technologie byly vhodné pro daný záměr.

2 Prostředí Windows 8

Operační systém Windows 8 přišel do prodeje 26. října 2012. Je nástupcem velice úspěšného operačního systému Windows 7. Byl vydán ve verzích jak pro 32 tak i pro 64 bitové procesory. Je to první, klasický operační systém společnosti Microsoft, byl vydán i pro procesory ARM. Tato verze je označována jako Windows RT a je mířenou pro mobilní zařízení, především pro tablety. Bohužel aplikace vyvinuté pro 32 a 64 bitovou verzi, nejsou dostupné pro RT verzi. Mobilní a klasická verze je však propojena novým prostředím Modern UI, pro které mohou společné aplikace vznikat.

2.1 Modern UI

Modern UI je nové prostředí společnosti Microsoft vystupující ve Windows 8. Zaměřuje se na typografii a obsah v jednoduchém, minimalistickém pojetí. Inspiraci čerpá z moderních designových směrů jako je Bauhaus, Swiss Style a Motion Design. Všechny tyto směry spojuje čistota, použitelnost a dobrá čitelnost, odstraňuje zbytečné a rušivé elementy. Snaží se vytvořit více z mála a zaměřit se na aktuálně obsahově důležité detaily pro uživatele. Důležitou vlastností je přizpůsobení pro použití na dotekových zařízeních, s různými velikostmi obrazovek.

Obrázek 1: Modern UI Windows 8

2.1.1 Historie Modern UI

První náznaky tohoto prostředí můžeme najít už ve Windows 7, ale také starších projektech jako je Windows Media Center nebo Zune. Dnešní podoba se objevila poprvé až s Windows Phone 7.

Původní jméno tohoto prostředí je Metro. V srpnu 2012 bylo oznámeno ukončení využívaní interního jména Metro novým. Podle spekulací však tato změna mohla být ze snahy předejít právnickým sporům s německou společností Metro Ag, vlastníci ochrannou známku.

2.2 Windows Store apps

S novým prostředím a možností tvořit aplikace pro společné použití a také v rámci trendu poskytovat aplikace pro svou platformu na jednom místě, přibyla služba Windows Store. Jedná se o internetový obchod s aplikacemi, usnadňující získávaní programů oproti klasické cestě přes krabicové verze nebo samostatné výrobce. V současné době jsou v nabídce aplikace určené pouze pro prostředí Modern UI. Do budoucna se počítá s propojením s obchodem pro Xbox a rozšíření obchodu o další aplikace, zvláště pro aplikace pro klasický desktop.

2.3 Možnosti vývoje

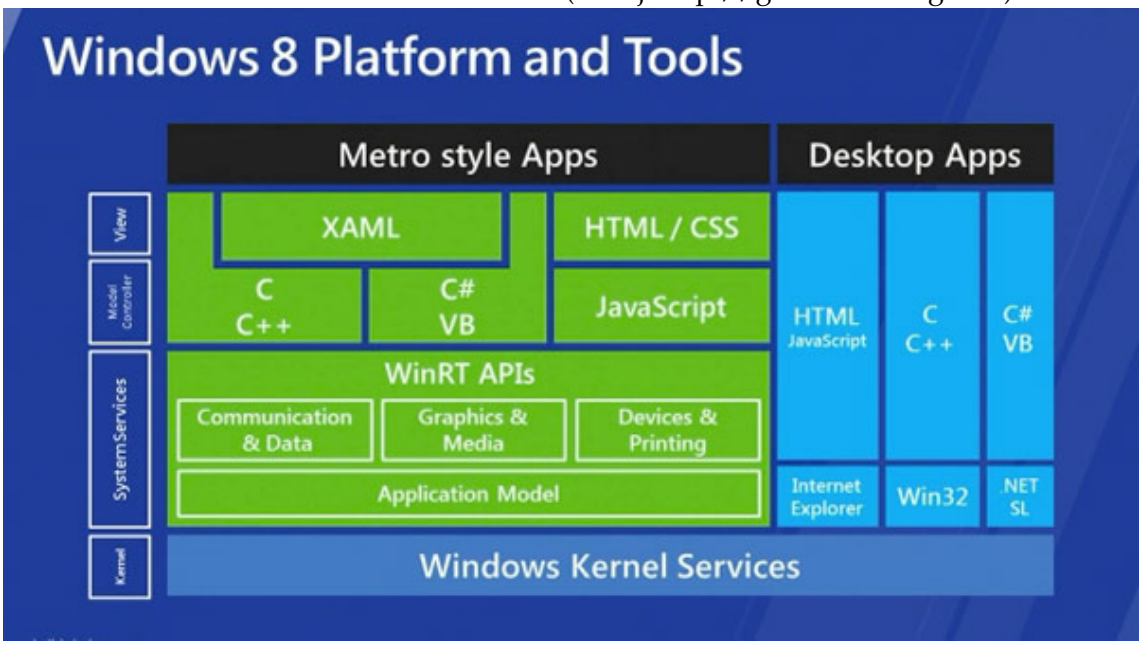
V současné době je Windows 8.1 nejnovější verze operačního systému od firmy Microsoft. Tento operační systém nabízí prostředí pro vývoj moderních aplikací, které mohou být psány v rozdílných programovacích jazycích. Mezi klasické zástupce patří C++ nebo C#, upravené o nové možnosti prostředí.

Na obrázku2 jsou zobrazeny všechny dostupné technologie pro vývoj jak Modern UI (zde však ještě se starým označením Metro), tak i pro klasické verze. Zajímavostí může být rozdělení částí View u Modern UI aplikací oproti klasickým, kde se grafické prostředí vytváří stejným programovacím jazykem. Tento způsob nemusí být vždy ideální, protože se hůře odděluje logická část aplikace od části vzhledové.

Důležitou vlastností vytváření aplikací pro rozdílné velikosti zařízení, je co nejvíce zkrátit čas strávený na přípravě GUI (uživatelského prostředí). Je záměrem, aby uživatelské prostředí bylo jednou navrženo a při vývoji jednou implementováno se zohledněním responsivity. Tento trend vývoje můžeme pozorovat při vývoji webových aplikací, protože rozšiřování chytrých telefonů je rychlé a posouvá klasický web do pozadí. Hlavním cílem je tedy mít aplikaci použitelnou na každém zařízení.

Právě webový standard HTML5 se snaží být dobrým a moderním směrem pro vývoj a to i klasických aplikací. Další důvod, který by mohl byt zmíněn je multiplatformnost řešení. Právě z tohoto důvodu firma Microsoft přišla s možností vývoje aplikací pomocí webových technologií i ve svých produktech.

Obrázek 2: Platforma windows 8. (Zdroj: http://geekswithblogs.net)



2.4 Další verze po Windows 8

2.4.1 Windows 8.1

Necelý rok po vydání Windows 8, Microsoft uvolnil aktualizaci označenou jako 8.1. Byla vydána zdarma pro všechny uživatele. Tato aktualizace přinesla několik novinek v grafickém prostředí a optimalizaci výkonu. Nejhlavnější grafickou novinku byl návrat tlačítka Start, které již nepředstavuje nabídku známou z Windows 7 a starších, ale pouze přístup do prostředí Modern UI.

2.4.2 Windows 10

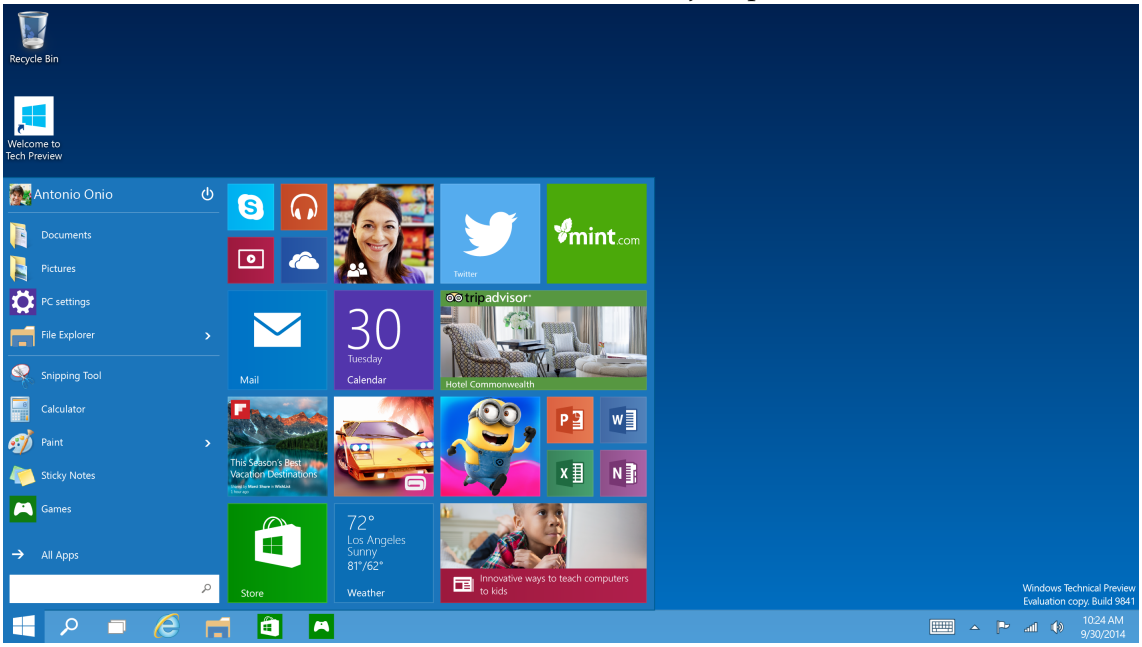
Následující verzí, která má přijít na trh na podzim roku 2015, je verze Windows 10. Touto verzí Microsoft chce sjednotit ještě více všechny dostupné zařízení k jednomu operačnímu systému, jednomu uživatelskému profilu pro každého uživatele. Míří také na svůj Xbox a do budoucna se možná dočkáme i chytrých nositelných věcí, jako jsou hodinky. Společnou vlastností všech zařízení bude tedy jádro operačního systému.

V této verzi se také vracejí některé rysy původních verzí, jako je již zmíněná nabídka Start, která nyní bude představovat zmenšené Modern UI, obrázek 3.

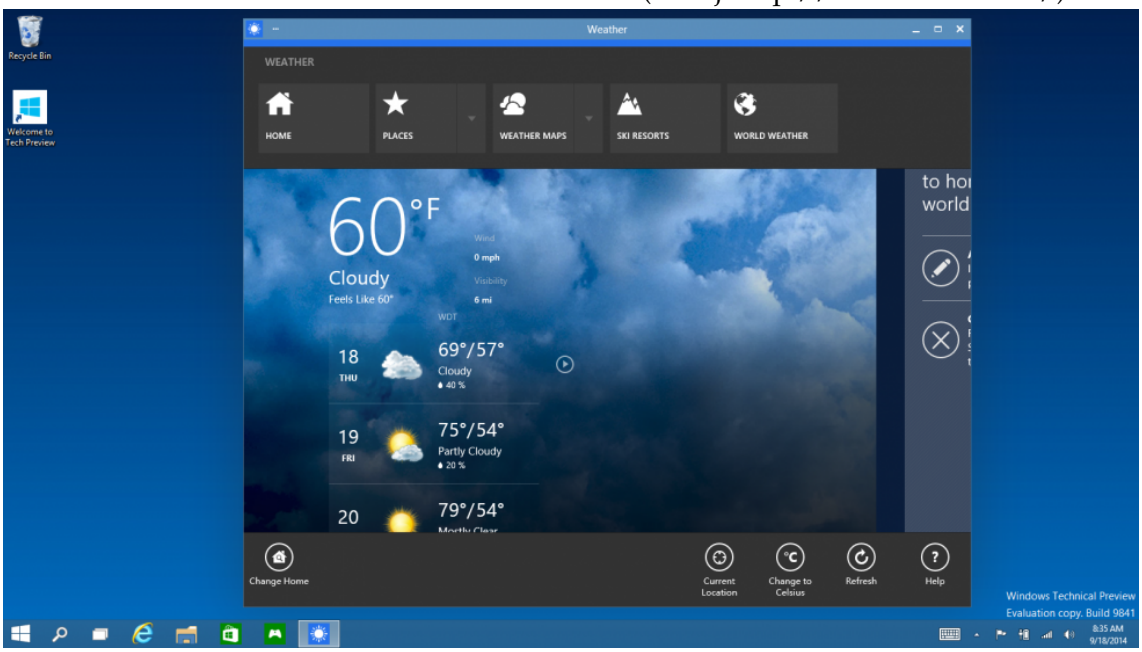
K propojení obou prostředí také napomůže možnost zmenšení oken aplikace z celoobrazovkového módu do módu okenního, obrázek 4.

Tyto informace nejsou konečné, finální vzhled a funkčnost se může ještě změnit. Rozhodně se máme na co těšit.

Obrázek 3: Windows 10 - nabídka Start. (Zdroj: http://arstechnica.com/)



Obrázek 4: Windows 10 - okenní zobrazení. (Zdroj: http://arstechnica.com/)



3 Microsoft WinJS

Technologie WinJS je prostředkem pro vývoj klasických aplikací tak také i mobilních, založených na webových technologiích. Je tvořena kombinace HTML, CSS a JavaScriptu, označovanou standardem HTML5.

WinJS je postavená jako Windows knihovna, což znamená, že obsahuje funkce pro práci s prostředím Windows. Implementuje obsluhování souborového systému, práci s klasickými senzory, které můžeme najít v zařízeních, jakým je třeba kamera, gyroskop, nebo GPS.

3.1 Vývoj WinJS

3.1.1 Verze 1.0

První verze WinJS přišla na trh s Windows 8 a dovolovala vytvářet aplikace jen pro počítačovou verzi Windows Store.

3.1.2 Verze 2.0

Další verzí Microsoft umožnil vytvářeních nativních aplikací pro Windows 8.x i Windows Phone 8.x. Přibyla také podpora pro Xbox aplikace.

3.1.3 Verze 3.0

Obecně předchozí verze WinJS umožňovaly vývoj jen pro operační systémy Microsoftu. Nová verze přinesla podporu také i pro další operační systémy a všechny hlavní prohlížeče dostupných pro tyto systémy. Dovoluje nám tedy vytvářet zcela multiplatformní aplikace napříč všemi dostupným zařízeními.

Jinými slovy, aplikace postavené na verzi 1.0 a 2.0 byly vyvíjeny ve Visual Studiu a určeny pro Windows Store, nová verze umožňují být platformě nezávislí a také nabízí svobodnou volbu vývojového prostředí.

Novou vlastností je modulárnost, která umožňuje mimo jiné i optimalizaci výkonu. Dovoluje načítat jen moduly, které jsou doopravdy v aplikaci využívány, čímž snižuje nároky na paměť.

3.1.4 Verze 4.0

V průběhu zpracovávání této práce, byla oznámena nová verze WinJS na vývojářském blogu Microsoftu [1].

Podle zprávy nová verze WinJS cílí opět na vylepšení podpory mezi prohlížeči, zároveň se snaží potvrdit fakt, že se nejedná o framework pouze pro aplikace, ale také pro

[1] Odkaz na článek https://blogs.windows.com/buildingapps/2015/03/27/a-preview-of-winjs-4-0/

web. Vývojáři se snaží přinést podporu také pro další frameworky, jako je například AngularJS[2], a další zajímavé funkčnosti. Součásti oznámení je také ukázka nové verze, se všemi novinkami, které přináší.

3.2 Visual Studio

Visual Studio je vývojové prostředí společnosti Microsoft, primárně určené pro vývoj aplikací pro Windows. Nabízí podporu pro všechny dostupné programovací jazyky, jak pro konzolové aplikace, tak i aplikace s grafickým prostředím. Visual Studio obsahuje několik prvků, prvním je samotný editor kódu, který podporuje zvýraznění syntaxe a automatické dokončování pro všechny programové prvky. Obsahuje také další funkce pro rychlejší orientaci v kódu a následné úpravy.

Visual Studio kompiluje kód na pozadí, aby poskytlo informace o syntaktických a kompilačních chybách a ty následně zvýrazňuje. Tato kompilace však nevytváří spustitelný kód, slouží pouze pro pohodlí programátora.

Další důležitou součástí je debugger, což je softwarový nástroj, který se používá pro hledání chyb při vývoji software ve fázi ladění. Debugger pracuje se jak spuštěným kódem ve strojové formě tak také nad spravovaným kódem, tímto způsobem ulehčuje hledání chyby.

Pro vývoj mobilních aplikací na osobním počítači neexistuje způsob, jak otestovat dotykové ovládání nebo využít specializované senzory, jakým je gyroskop nebo akcelerometr. Pro tento případ Visual Studio obsahuje také simulátor mobilního zařízení (viz. obrázek 5). Simuluje také dotykové akce pro různé velikosti obrazovek a rozlišení. Tento způsob není však jedinou možností testování aplikace, je možné využít běh na lokálním zařízení nebo využít reálného zařízení. Tato přímá podpora nabízí několik výhod oproti nainstalovaní aplikace klasickou cestou do zařízení a následném testování v tom, že dovoluje využití stejných ladících možností jako na lokálním zařízení. Důležitou a nespornou výhodou je také pohodlí.

Dalším nástrojem je pomocník pro návrh grafického prostředí. Hlavním cílem je umožnit jednoduchým přetažením vkládat grafické prvky na plochu a umisťovat je. Není nutné psát kód ručně, spouštět program a vizuální kontrolou se přesvědčovat o finálním umístění. Visual Studio nenabízí samo o sobě podporu pro návrh grafického prostředí pro WinJS, ale je obsaženo v nástroji jménem Blend for Visual Studio. Tento program umožňuje editaci CSS stylů, tak také návrh CSS animací.

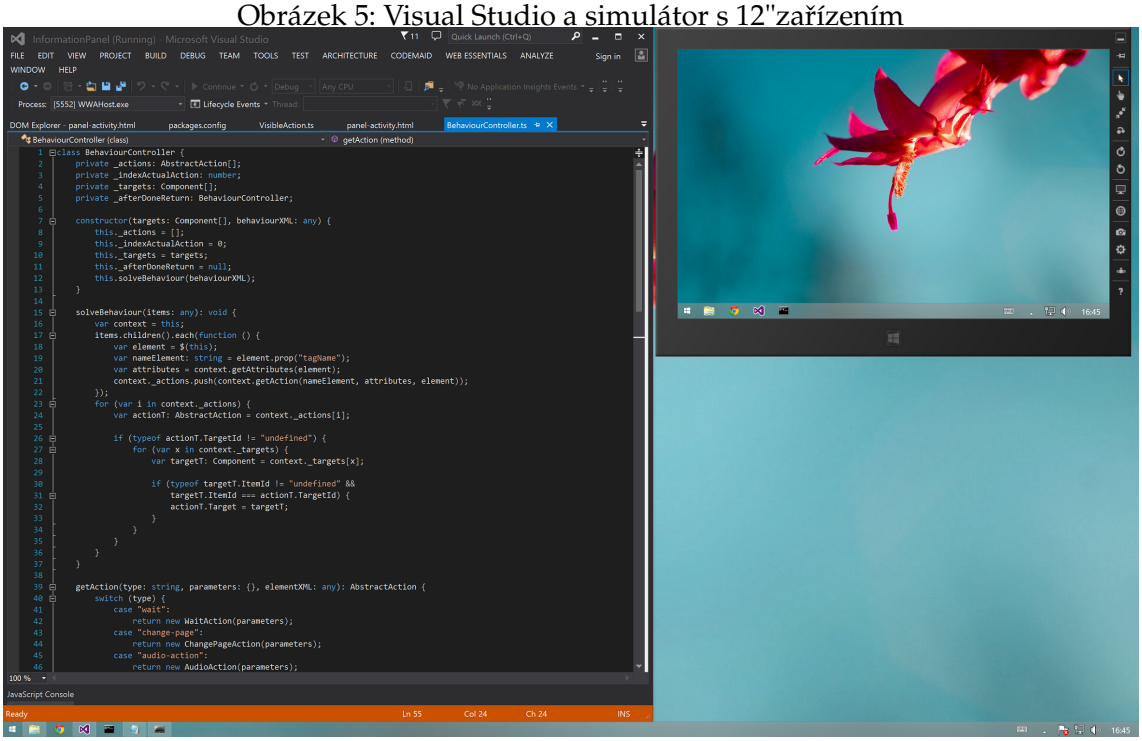
Visual Studio studio dále obsahuje nástroje pro databáze, správu práce v týmu a další nástroje mimo téma této práce. Důležitou vlastností je rozšiřitelnost o další nástroje nebo podporu programovacích jazyků.

3.2.1 Vytvoření projektu

WinJS je součástí Microsoft Visual Studia již od verze 2012. Existují však i další způsoby jak WinJS získat, jednou možností je stažení ze služby GitHub nebo využití běžných balíčkováních systému dostupných pro JavaScript.

[2]https://angularjs.org/

Obrázek 5: Visual Studio a simulátor s 12"zařízením



3.3 Omezení aplikace

Javascript obecně, je kód spouštěný v prohlížeči téměř bez vědomí uživatele (může být vypnutý, ale většina dnešních webů je potom nepoužitelná). Tato situace však vytváří nepříjemná bezpečnostní rizika, která musí být řešena na úrovni jádra prohlížečů.

Klasická Javascriptová aplikace nemá přímý přístup datům uživatele. Uživatel může poskytnout soubor, který je ale následně oddělen od zbytku souborového systému, schováním své cesty a až poté je předán ke zpracování. Bez této ochrany by kterýkoliv server mohl obsahovat útočný kód, zkoumající naše osobní data, nebo k nim přímo přistupovat a odesílat útočníkovi.

4 Konstrukční prvky

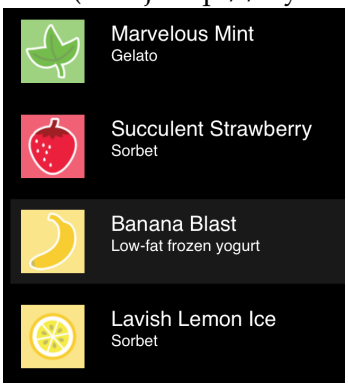
4.1 Vizuální prvky WinJS

V následující části je výběr zajímavých komponent, všechny jsou popsány v dokumentaci.

4.1.1 Seznam - ListView

Touto komponentou se snadno vytvářejí seznamy položek. Každá položka může být jednoduše přizpůsobitelná, vlastním rozmístěním. Seznam disponuje dostatkem zajímavých funkcí, jako je třeba sloučením položek do jedné části - například pro abecední seznamy. Seznamy nemusejí být řazeny jen klasicky pod sebou ale také do stran nebo mřížky. Není problém přidat hlavičku, anebo patičku.

Obrázek 6: Seznam (Zdroj: http://try.buildwinjs.com/)



Velice zajímavým doplňkem této komponenty je, že již v základu obsahuje možnost uživatelského řazení položek klasickým gestem táhni a pusť. Velice přínosným může být líné načítaní (další data načítá, až jsou potřeba například, když se uživatel blíží konci dostupných položek, začne načítat další).

4.1.2 Galerie - Flip View

Komponenta galerie slouží pro zobrazování kolekce obrázků, jednoho obrázku v jeden čas. Tato komponenta obsahuje nejnutnější nastavení, například automatické posouvání, zvolení animace výměny obrázku, tabulku obsahu.

4.1.3 Animace

ModernUI má, kromě svého specifického vzhledu, své specifické animace. Příkladem jednotlivých animací může být přecházení mezi stránkami aplikace, změna textu a atd. Veškeré tyto animace jsou nachystány pro jednoduché použití v každé aplikaci.

Obrázek 7: Flip view (Zdroj: http://try.buildwinjs.com/)

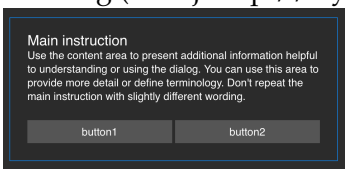
4.1.4 Flyout

Komponenta představující kontextové menu nebo seznam akcí pro tlačítko, a je také známá jako "dropdown".

4.1.5 Dialogové okno Content Dialog

Tato komponenta vynutí pozornost uživatele při práci s aplikací, může představovat chybovou hlášku, ale také může být použita jako potvrzovací formulář k dané akci.

Obrázek 8: Content Dialog (Zdroj: http://try.buildwinjs.com/)



4.1.6 Fragments

Komponenta fragment představuje část uživatelského prostředí, fragmenty do prostředí můžeme přidávat dynamicky, přitom zbylý obsah může zůstat stejný. Fragmenty se využívají také, pokud se část rozhraní opakuje.

4.1.7 Výběr souborů - File Picker

Komponenta výběr souboru nemá svůj vzhled dokud není otevřena, poté nabízí rozhraní pro přístup k datům uživatele. Tento výběr je však v režii operačního systému.

4.2 Programové prvky WinJS

4.2.1 Práce se souborovým systémem

WinJS poskytuje pro potřeby programátora lokální oddělený adresář aplikace, kde může uložit soubory trvale a poté k nim přistupovat. Tento způsob je rozdělen do několika typů, které se liší dobou uchovávání a způsobem synchronizace s dalšími zařízeními. O tom, který typ využije, rozhoduje přímo programátor.

1. localSettings - automaticky uložené nastavení aplikace definované jako klíč, hodnota
2. roamingSettings - podobně jako localSettings, jen s rozdílem že toto nastavení se může synchronizovat napříč zařízeními
3. temporaryFolder - místo pro ukládání dočasných souborů, u těchto souborů není zaručeno, že jsou trvalé, mohou být kdykoliv smazány systémem
4. localFolder - místo pro ukládání jakýchkoliv souborů, zaručena trvanlivost, nesynchronizují se napříč zařízeními
5. roamingFolder - místo pro ukládání dočasných souborů, je nutno brát ohled na to, že tyto soubory jsou synchronizovány

Umístění adresáře je v lokální složce profilu uživatele, v této složce sídlí nastavení většiny ostatních programů. Každou aplikaci představuje jeden adresář. Soubory v lokálním adresáři je možno přesouvat, mazat, číst a také zapisovat.

Následuje ukázka kódu pro otevření souboru *info.txt* z dočasné složky aplikace a jeho následné přečtení do proměnné *lines*, která je předána jako parametr zanořené funkce.

```
Windows.Storage.ApplicationData.current.temporaryFolder.getFileAsync("info.txt")
    .then(function ( file ) {
            Windows.Storage.FileIO.readTextAsync(file)
                .then(function(lines) {
                    // function body
                }
        );
    });
```

Výpis 1: Otevření a přečtení souboru info.txt z adresáře aplikace pro dočasné soubory

4.2.2 Promisses

Většina událostí v Javascriptu se děje asynchronně oproti běžnému toku, dobrým příkladem může posloužit dotazování na server. Je uskutečněno asynchronním požadavkem, kterým se táže serveru a čeká na jeho odpověď. Pokud by tento dotaz čekal na odpověď a nedělal nic jiného, aplikace by byla po tuto dobu neaktivní. Odpověď je bezodkladně obsloužena po příchodu odpovědi.

Asynchronní události jsou řešeny callbacky, což jsou funkce, které se volají po dokončení události - v našem případě se dotaz na server dočkal odpovědi. Parametry těchto funkcí představují různé typy odpovědí.

```
function loadXMLDoc()
{
    var xmlhttp;
    if (window.XMLHttpRequest){
        xmlhttp=new XMLHttpRequest();
    }else{
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }

    xmlhttp.onreadystatechange=function(){
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            // Uspesne ziskani souboru
        }else{
            // Neuspesne ziskani souboru
        }
     }

    xmlhttp.open("GET","ajax_info.json", true);
    xmlhttp.send();
}
```

Výpis 2: Ukázka klasického XHRT

Tento způsob se však stává dost nepřehledným, pokud těchto událostí je nutno provést sekvenčně za sebou více a další dotazy jsou datově závislé na předchozím. V následující ukázce je možno si také povšimnout úspory kódu, při stejně funkčnosti.

```
WinJS.xhr({
    url : rssURL,
    responseType: "document"
    }) .done(
        function completed(result) {
            // Uspesne ziskani souboru
        },
        function error () {
            // Neuspesne ziskani souboru
        }
    );
```

Výpis 3: Stejný požadavek v prostředí WinJS

4.2.3 Plánovač

Plánovač představuje manažera pro asynchronní úkoly, dovoluje určovat prioritu zpracovávaného úkolu, pozastavovat a znovu se navracet ke zpracovávání.

Prioritu zpracovávaného úkolu je možno určit číslem -15 až 15, kde menší hodnota znamená nízkou prioritu. Základní prioritou je 0, značící normální stav.

1. -15 Min
2. -13 Idle
3. -9 Below normal
4. 0 Normal
5. 9 Above normal
6. 13 Hight
7. 15 Max

```
var S = WinJS.Utilities.Scheduler;

S.schedule(function () {
    // function body
}, S.Priority.high);
```

Výpis 4: Spuštění asynchronního úkolu s vysokou prioritou

Dlouho trvající funkce mohou jednoduše pozastavit své vykonávání, aby uvolnily místo pro úkoly s vyšší prioritou. Aby řízení úkolů bylo kompletní, manažer disponuje také funkcemi pro zastavení vykonávání úkolu s menší prioritou, než mu je zadána.

4.3 TypeScript

TypeScript je nástavba k JavaScriptu, rozšiřující vývoj aplikací o výhody, jakou je typová kontrola, jednodušší zápis tříd a modulů. Zmíněné výhody nejsou nápomocny jen při vývoji, ale také při pozdějších úpravách a ladění.

TypeScript nemá svůj vlastní interpretr jazyka, ale je transpilován do JavaScriptu. Transpilace je proces, při kterém se ze zdrojového kódu stává zase zdrojový kód, oproti komplikaci, kdy vzniká strojový kód. Obecně platí, že výsledný kód je podmnožinou kódu zdrojového, tady platí, že každý JavaScript je TypeScript.

4.3.1 Vývoj v TypeScriptu

TypeScript je funkčně objektový skriptovací programovací jazyk. Jedná se o rozšíření Javascriptu, nesnaží se jej měnit oproti ostatním projektům, jako je například CoffeScript nebo Script#. Snaží se změnit způsob zápisu do do přehlednější varianty.

Co tedy Typescript nabízí? Statické datové typy, třídy a dědičnost, moduly, rozhraní, generické datové typy a výchozí hodnoty parametrů funkcí. Všechny prvky jsou však dosažitelné i bez jeho použití, ale za cenu složitějších konstrukcí.

Následující útržek zdrojového kódu představuje zápis třídy, konstruktoru, instančních proměnných a funkci v TypeScriptu. Poté následuje transpilovaný výstup v Javascriptu. Na tomto příkladu je zřejmé, jak TypeScript zjednodušuje zápis a do jaké míry mění výsledný kód.

```
class ExampleClass {
    public variable1: number;
    protected variable2: string;
    private variable3: number[];

    constructor() {
        this.variable1 = 42;
        this.variable2 = "Test";
        this.variable3 = [];
        this.variable3.push(42);
    }

    public setVariable(num: number) {
        this.variable1 = num;
    }

    public getVariable1(): number {
        return this.variable1;
    }

    protected isNumberInArray(num: number): boolean {
        return num in this.variable3;
    }

    protected addToText(text: string) {
        this.variable2 += text;
    }
}
```

Výpis 5: Ukázka definice třídy v TypeScriptu

```
var ExampleClass = (function () {
    function ExampleClass() {
        this.variable1 = 42;
        this.variable2 = "Test";
        this.variable3 = [];
        this.variable3.push(42);
    }

    ExampleClass.prototype.setVariable = function (num) {
        this.variable1 = num;
    };

    ExampleClass.prototype.getVariable1 = function () {
        return this.variable1;
    };

    ExampleClass.prototype.isNumberInArray = function (num) {
        return num in this.variable3;
    };

    ExampleClass.prototype.addToText = function (text) {
        this.variable2 += text;
```

```
    };

    return ExampleClass;
}) ();
//# sourceMappingURL=BachelorTest.js.map
```

Výpis 6: Kód třídy TypeScriptu transpilovaný do JavaScriptu

4.3.2 Typová kontrola

Typová kontrola kódu probíhá při transpilaci do JavaScriptu, Visual Studio umí však hlásit chyby o špatném typu proměnné i v reálném čase při vývoji. Typová kontrola probíhá jen při překladu, poté se zahazuje. Tímto způsobem se neubírá rychlost při vykonávání hotového skriptu.

Pro využívané projekty, například knihovny, je nutno dodat soubor s deklaracemi funkcí. Výhodou je, že velká část používaných knihoven má své deklarace připravené k použití.

4.3.3 Dědičnost

V následující kapitole rozšíříme ukázku zápisu třídy o dědičnost. V tomto případě probíhá jediné přidávání dalšího kódu do výstupu. Samotná dědičnost funguje jako v klasických objektově orientovaných programovacích jazycích.

```
class BaseExampleClass {
    public variable1: number = 42;
}

class ExampleClass extends BaseExampleClass {
    public variable2: number;

    constructor() {
        super();

        this.variable2 = 42;
    }
}
```

Výpis 7: Dědičnost v TypeScriptu

```
var __extends = this.__extends || function (d, b) {
    for (var p in b) if (b.hasOwnProperty(p)) d[p] = b[p];
    function __() { this.constructor = d; }
    __.prototype = b.prototype;
    d.prototype = new __();
};
var BaseExampleClass = (function () {
    function BaseExampleClass() {
        this.variable1 = 42;
    }
```

```
    return BaseExampleClass;
}) ();
var ExampleClass = (function (_super) {
    __extends(ExampleClass, _super);
    function ExampleClass() {
        _super.call(this);
        this.variable2 = 42;
    }
    return ExampleClass;
}) (BaseExampleClass);
//# sourceMappingURL=BachelorTest.js.ma
```

Výpis 8: Výstup z ukázkového příkladu s dědičností

4.4 Less

Less je preprocesor pro kaskádové styly, rozšiřuje CSS o následující dynamické prvky:

1. Proměnné
2. Skládání stylů (Mixins)
3. Vnořená pravidla
4. Funkce
5. Výpočty

Less svými funkcemi napomáhá k redukci duplicitního kódu. Pro každou třídu, která by měla obsahovat stejné vlastnosti, je možné vytvořit jednu třídu a postupným skládáním se dostat k finálnímu výsledku. Pro příklad máme třídu, která se stará o přidání okraje bloku a k tomu více různých tříd bloků, u kterých bychom tento okraj použili. Místo kopírování těchto vlastností můžeme tuto třídu přiřadit třídě bloků a vlastnosti budou použity. Následná změna okraje se týká jednoho místa v kódu. Tento problém se dá řešit i způsobem spojení vlastností až v HTML, ale obecně může být některá třída zapomenuta, popřípadě při více třídách se tento způsob již stává dosti nepřehledným.

Vlastnosti pro skládání tříd mohou být ovlivněny i parametry nebo přidáním pravidla podle nějaké podmínky, podobně jako jsou *media queries* v čistém CSS, ale s rozdílem, že parametrů, které můžeme použít je daleko více.

Přínosnou vlastností je použití proměnných, do kterých může být uloženo číslo, hodnota nebo barva. Následná změna hodnoty v celém dokumentu znamená přepsání jednoho literálu a zkompilování. Většinou se však v celém dokumentu nepoužívá jedna barva, ale skupina barev, která ze stylistického hlediska má většinou podobné vlastnosti. Může jít o barvu světlejší, nebo sytější a podobně, z tohoto důvodu Less disponuje funkcemi pro úpravy barev.

Čísla v proměnných zase mohou být použity pro všechny škálovatelné vlastnosti CSS, jako je např. výška, šířka, tloušťka ohraničení. Pro číselné konstanty jsou zde také matematické funkce. Dalším využitím je uložení přímé vlastnosti, např. pro okraje solid, dashed a podobně.

5 Informační panel

V současném světě narážíme na informační panely téměř na každém místě. Může jít jak o tabule s turisticky nebo historickými zajímavými informacemi o místě kde se právě nacházíte, také třeba podnicích, jako jsou obchody, kde můžou vyobrazovat nabízené produkty, výrobky v akčních cenách nebo třeba informovat o nabízených službách. Informačních panelů, tvořenými klasickými tištěnými cedulemi je nespočet. Cílem je tedy navrhnout řešení, které by je mohlo nahradit modernější technikou.

Aktuálně je dostatek technologií k použití k těmto účelům. Nemusí jít nutně o dedikovaná zařízení mířená k jedinému účelu, ale je možno využít klasických počítačů ve spojení s televizí, či velkoplošných barevných LED zobrazovačů. Díky těmto zařízením se může výrobní cena snížit na nezbytné minimum, protože není nutné vytvářet zařízení jen pro jeden účel. Nyní se již dostáváme k situaci, kdy v technologii můžeme mít jasno, ale chybí nám způsob zobrazení obsahu. Jednoduchým řešením by se mohlo zdát, promítat video nebo obrázky připravené na míru. Tato strategie se může zdát výhodnou, protože možnosti jsou prakticky neomezené. Ale v obou případech jsou následné úpravy složité. Změna takového média by znamenala změnit celý záznam. Ne vždy však můžeme mít po čase přístup k editovatelným souborům.

Nápad jak porazit předešlé problémy, je vývoj aplikace, která by byla vyvinuta speciálně pro jeden případ použití. V potaz je brána situace, kdy jednotlivé případy se mezi sebou liší pouze finálním vzhledem, ale logické jádro celé aplikace může být založeno na společných vlastnostech. Jde o velmi přímočaré řešení, trh nabízených technologií využitelný pro aplikace je velice široký.

Vývoj specializované aplikace pro jeden případ použití, může stále být optimální pro velmi specifické požadavky, obecně je ale třeba se zaměřit na větší spektrum případů. Nápadem je tedy aplikace vyvinuta s cílem umět si nějakým způsobem poradit s většinovým případem použití.

V této situaci tedy nastává otázka, proč tedy takovou aplikaci vyvíjet? Jakým způsobem zajistit takové použití? A jakou přidanou hodnotu by měla přinést? Odpověď již možná zazněla v předchozích odstavcích, cílem je možnost tvořit dynamický obsah a usnadnit správu aktualizací při změnách předávaných informací. Velmi přínosnou vlastností je také schopnost vytvořit prezentaci bez znalostí jakýchkoliv technologií spojených pokročilejší znalostí počítače.

5.1 Návrh aplikace

Při vývoji aplikace musí být kladen co nejvyšší důraz na možnosti použití pro jakoukoliv situaci. Další silnou částí, která by měla být zohledněna, je možnost budoucí rozšiřovatelnosti. V základním tvaru by tedy měla byt schopna uspokojit co nejvíce zákazníku svou funkčností a konfigurovatelností. Současné technologie a solidní návrh bude nápomocen tyto předpoklady splnit.

Prvotním cílem aplikace je splnit veškeré vlastnosti, které mohou být vytvořeny klasickým informačním panelem. Tento případ obsahuje pouze dvě komponenty, jsou jimi text a obrázky. Komponentou je myšlen prvek, obalující jeden druh obsahu, dalším pří-

kladem může být video, zvuk, nebo speciální stavy jiných komponent (např. komponenta zobrazující aktuální čas).

Krokem k dynamičnosti, je nutno definovat jazyk, kterým administrátor (člověk provozující informační panel, nebo osoba pověřená) předá aplikaci svůj obsah.

5.1.1 Požadavky

Nejdůležitější požadavky již byly vyjmenovaný. V této části se již dostáváme ke specifičtějším vlastnostem aplikace. Předpokládáme tedy aplikaci, která bude přijímat vstup na základě, kterého bude zobrazen obsah a vykonávat nějakou funkčnost.

Vstup bude obsahovat komponenty definované svými vlastnostmi, obsahem a vzhledem pro daný případ, každá komponenta tedy musí být dostatečným způsobem konfigurovatelná. Nejenom komponenty, nýbrž celá aplikace musí být schopna být upravitelná k jakémukoliv výsledku. Vzhled výstupu by měl být směrem k zaujmutí pozorovatele, k upoutání jeho zvědavostí a předání informací.

Součástí bude také již zmíněný scénář akcí, který by měl být pevně svázán s komponentami. Pokud součást má být definována svým obsahem, je důležité moci tento obsah dynamicky měnit. Každá komponenta může mít také své specifické akce.

Předpokladem je, že obsah informačního panelu je promítán cyklicky. Zároveň délka jednoho cyklu by neměla být příliš dlouhá, ale měla by předat dostatek informací, a přitom držet stálou pozornost. Očekávaným provozem aplikace bude režim, kdy informační panel vykonává činnost bez zásahu pozorovatele. Tato činnost bude ovládaná částí vstupu starající se o chování panelu, tvořenými jednoduchými akcemi s jasnou konstrukcí. Typickým příkladem akce může být třeba situace změny textu, obrázku a podobně.

Scénář chování by měl být ovlivnitelný podmínkami řídící tok akcí. K povaze použití informačních panelů je možné připravit již dané rozmezí podmínek, naplňující různé použití. Podmínkou k zobrazení určitého obsahu může být čas, datum nebo třeba aktuální den. U těchto podmínek není nutné vyvíjet úplnou komplexitu na všechny možné stavy, ale spíše vést tuto část k co největší jednoduchosti. I přesto je možností jakými způsoby ovlivňovat akce je dostatek a budou se více otevírat s přidáváním funkčností aplikace.

Mimo popsané autonomní chování by i pozorovatel mohl ovlivnit akce informačního panelu. Může se jednat o stisk nějaké klávesy nebo dotyk na obrazovce. Všechny tyto události jsou technicky řešitelné na běžných zařízeních. Zajímavou vlastností může být akce na přistoupení pozorovatele blíže k obrazovce, tato akce je zaznamenatelná pomocí web kamery vyhodnocující změnu sledovaného prostředí.

Narušením autonomního chování je možno reagovat na preference uživatele, kdy nemusí jít o přímé převzetí kontroly, ale jen o úpravu parametrů zobrazení, jako je třeba přeskočení na jinou část obsahu nebo jen urychlení aktuální části. Po daném intervalu se chování vrací do běžného chování. Dalším příkladem může být změna lokalizace obsahu, která ovlivní všechen obsah informačního panelu. Tato změna však může zůstat trvalá do dalšího přepnutí.

5.2 Komponenty

Jak již bylo zmíněno, prostředí informačního panelu bude sestaveno z komponent. Tato část bude tedy orientovaná k podrobnějšímu popisu navrhovaných komponent.

5.2.1 Komponenta pro text

Základním kamenem informačního panelu jsou textová informace. Tato komponenta se může tvářit staticky, ale je důležité, aby byla ovlivnitelná již zmíněnými akcemi, v tomto případě budou manipulovat s obsahem. Nemusí jít jen o prostou změnu, ale i o možnost rolovat text pokud je dlouhý a další podobné animace.

Tato komponenta nebude zobrazovat jen zadaný text, ale také texty podle různých šablon, jako je čas, datum. Tyto šablony by měly být konfigurovatelné jednoduše pro každý formát data a času. K těmto šablonám může přibýt zobrazení svátku, jména dne a podobně.

Informační panel by měl umět také načítat textový obsah různých dokumentů běžných formátů.

5.2.2 Komponenta obrázek

Další komponentou informačního panelu je obrázek. Tato komponenta by měla souviset s komponentou pro klasický text a měla by jej doplňovat. Akce pro změnu textu, může následovat akce změnění obrázku.

5.2.3 Galerie

Galerie je speciálním případem komponenty pro obrázek, musí umět sama nezávisle vyměňovat obrázky s nastavitelným intervalem, součástí by mělo být i více efektů výměny obrázků. Výhodou je však automatizace práce, kdy nebude nutná akce pro každou výměnu obrázku.

5.2.4 Přehrávač videí a audia

Přehrávač videí by měl umožnit jednoduché rozhraní pro přidání multimediálních prvků. Ideální se jeví podpora co nejvíce možných formátů. I tento prvek by měl být konfigurovatelný a akcemi by měl být dostatečně ovladatelný. Rozhodně by neměly chybět akce pro spuštění, zastavení, změnu aktuálního času, nastavení hlasitosti s možností naprostého utišení.

5.2.5 Dynamické komponenty

Tato kategorie komponent bude zastřešovat komponenty, kterým obsah bude přidáván z nějaké služby, v konfiguraci bude tudíž jen odkaz či popis, říkající odkud svůj obsah mají získávat. Společným zdrojem pro tyto komponenty bude internet, existuje spousta zdrojů, které mají veřejně nebo licencí chráněné API, kterým je možno získávat zajímavé

informace, jako je třeba počasí, zprávy anebo jen galerie fotografií. Pro obecné zástupce používaných služeb, budou připlaveny komponenty, zajišťující komunikaci, získaní informace a následné zobrazení.

Za výběrem zde popsaných komponent stojí úvaha několika případu použití, obsahem informačního panelu nemusí být nutně jen informace zaměřené přímo pro dané použití, ale také doplňkové informace. Příkladem by mohla být turistická chata s informacemi o historii chaty, nabídkou služeb, ceníkem, obrázky nabízených pokojů a podobně. Tyto informace by měly být tedy tím nejdůležitějším, zároveň ale jistým lákadlem, proč se zrovna tady ubytovat, jsou v tomto případě ohlasy jiných návštěvníku. Tyto informace se nemusí získávat v rámci této aplikace, ale mohou být přístupné na službách a tímto se dostáváme k sociálním sítím. Sociální sítě nabízejí ideální prostor pro popsanou situaci a jsou také za tímto účelem využívány. Dalším nápadem pro horskou turistickou chatu může být předpověď počasí, zprávy z okolí a podobně.

U obsahu a míry konfigurovatelnosti u těchto komponent musí být přihlédnuto k implementaci přímých zdrojů dat. Obecně se dá říct, co komponenta, to poskytovatel služby nebo skupina služeb s velmi podobným výstupem.

5.2.5.1 Počasí Komponenta počasí by měla být spojující pro získávání dat ze zdrojů počasí jak pro aktuální počasí, tak také pro předpověď. Z vlastního principu věci je jasné, že tento obsah závisí na místě, pro které je informační panel provozován. Zdrojem této informace může být zařízení, které může nabídnout přesnou, anebo přibližnou polohu v závislosti na vybavenosti.

Komponenta by měla umožnit obsah informace, v některých případech nás může zajímat jen současná teplota, jindy klíčovým se může stát tlak a podobně. Počasí nemusí být nutně jen textová záležitost informací o aktuální teplotě, ale také může být vylepšená grafickými prvky, symboly a podobně. Tyto obrázky by měly být součástí komponenty s možností změn.

Akcemi této komponenty se nabízí vynucení získání nových dat, změna místa, nebo změna zdroje dat, pokud bychom chtěli přihlédnout k rozdílným předpovědím jiných služeb.

Tato komponenta by mohla být také zdrojem pro ovlivňování akcí informačního panelu. Obsah by tak díky těmto podmínkám mohl být zobrazován jiný pro teplé a studené dny, také by jej mohlo ovlivnit aktuální počasí. S příkladem turistické chaty by nemusel tedy doporučovat návštěvu koupaliště v deštivých dnech, ale třeba muzea.

5.2.5.2 RSS čtečka Standart RSS již není využívaný v tak velké míře, jako před lety. Je navržen k rychlému přístupu novinek z aplikací, starající se o více zdrojů současně. V hlavní míře jde o zdroje se zprávami, skládá se z nadpisu, většinou jen části informace a odkazem pro přístup k celé zprávě. Tyto části mají standardní tvar, tímto faktem je možno tuto komponentu abstrahovat až do úplné nezávislosti na konkrétním zdroji.

Povinným parametrem pro tuto komponentu v tomto případě je adresa zdroje, dalšími by bylo vhodné ovlivnit již zobrazované zprávy, omezit zobrazovaný počet, nebo je filtrovat.

Spektrum akcí je velice obdobné jako u komponenty pro počasí, může nabízet tedy vynucení aktualizace či změnu zdroje dat.

5.2.5.3 YouTube Přehrávač videí je zaměřen na videa uložená v zařízení a rozsah možných formátů je omezen pouze na závislost prostředí. Předpokládá se použití již nativního přehrávače, který by měl být přizpůsoben pro použití s daným vývojovým prostředím.

Komponenta by měla zastřešit získání videa z youtube.com a nad ním připravit stejné akce jako u klasického přehrávače.

5.2.5.4 Facebook, Twitter V úvodu této kapitoly byly zmíněny sociální služby, kterých je nepřeberné množství pro různě zaměřené použití. Facebook a Twitter patří mezi nejvíce oblíbené služby. Disponují velice dobře zdokumentovaným API.

5.2.5.5 Vzdáleně ovladatelná komponenta Nejspeciálnější komponentou by se mohla jevit taková, které obsah bude dodávat na dálku přímo administrátor, nebo jiný software starající se o tuto službu. Touto komponentou se již informační panel dá využít k daleko speciálnějším akcím, jako jsou například vyvolávací tabule na úřadech.

Princip této komponenty je prostý, jako u ostatních se komponenta bude dotazovat na nějakou službu o data, ale v kratším intervalu. Tento interval se může lišit přesnou situací, je přímo závislý na tom, jak se často se data budou měnit. Výstupem této služby budou již zmíněné akce informačního panelu, které budou dynamicky aplikované přímo. U těchto akcí se nedá již hovořit o opakovatelnosti informačního panelu, ale o speciálním použití. Opakovatelné vlastnosti, je nutno zachovat pro klasicky konfigurované prvky informačního panelu.

Díky této komponentě, je možné také zajistit situace, kdy konfigurace celého informačního panelů nebude součástí zařízení, ale může být vzdálená. S každým startem aplikace by tato konfigurace byla stažená a používaná, v této situaci je opakovatelnost akcí v čase splnitelná. Tato funkce tedy usnadní ovládání více stejných panelu na dálku, což může být také velice přínosná vlastnost.

5.2.5.6 Další komponenty Další komponenty jsou si dost podobné, nebo rozvíjejí vlastnosti jiných komponent, proto budou uvedeny jen navrhované názvy. Cíl těchto komponent by měl byt zřejmý, není nutné je implementovat hned v prvních verzích aplikace, mohou být dodány postupně. U těchto komponent se očekává méně časté použití než u výše zmíněných.

- Internetový prohlížeč
- Katalog produktů
- Internetové rádia
- Mapy

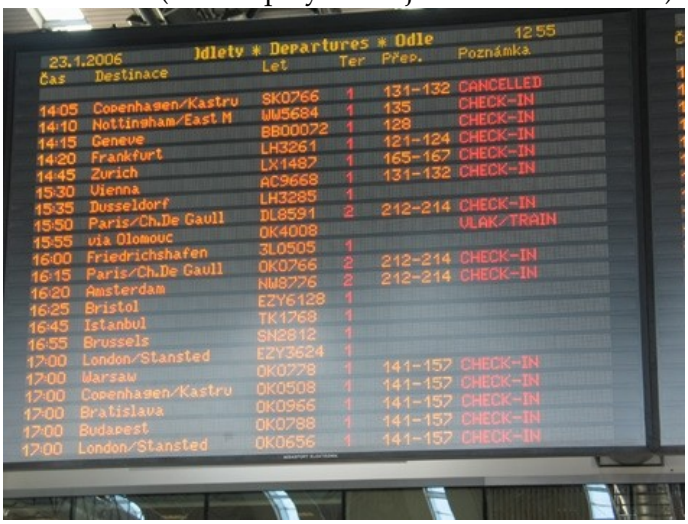
U komponent, které nemají přímou kontrolu nad obsahem, může nastat problém se zadanými požadavky nad obsahovou částí, typickým problémem může byt lokalizace do jiných jazyků. Některé zdroje mohou nabízet již lokalizované data. Obecně tato situace bude platit spíše u dat, která nabývají určitých hodnot, příkladem může být předpověď počasí, pro tento typ dat se dají velice jednoduše připravit slovníky s překlady. Dalším řešením může být implementace možnosti využít překladače třetí strany, umožňující překlady jakýkoliv dat, bohužel toto řešení nemusí být dostatečně kvalitní.

5.3 Konkurenční a podobné produkty

V rámci této práce je také důležité se poohlédnout, jaké možnosti jsou u konkurenčních produktů. V České republice se nachází několik firem, které se zabývají podobnými produkty. Většina těchto firem nabízí již kompletní řešení obsahující vlastní aplikaci a zobrazovací zařízení.

Většina nalezených firem nabízí informační panely tvořené LED obrazovkami, které umí zobrazovat jen text. Zobrazovače jsou tvořeny LED panely, které umí zobrazit znaky nebo jen čísla. Pokud je třeba neměnného textu, tak je vytvořen tiskem. Jejich použití můžeme najít například ve výrobních halách s přehledem vyrobených kusů, zmetků a podobně, skóre tabule ve sportovních halách, nebo také ve veřejné dopravě s informacemi o lince a podobně. Výhodou těchto LED zobrazovačů je možná lepší čitelnost z větší vzdálenosti a teoreticky menší elektrická spotřeba. Tyto parametry jsou však dostatečně závislé na prostředí a použití.

Obrázek 9: (Led displaye Zdroj: www.nowatron.cz)



Další skupinou, jsou firmy nabízející grafické LED zobrazovače. Jejich hlavní výhodou jsou obrovské úhlopříčky a pozorovací vzdáleností několik desítek metrů. Většinou je možné je připojit k osobnímu počítači jako další monitor a promítat na tyto panely vlastní obsah. Tyto zobrazovače se mohou použít i s navrhovanou aplikací.

Firem, které v České Republice existují s podobným produktem je velmi málo. Z důvodu málo výmluvných webových prezentací, vybírám ke srovnání firmu Ki-Wi, která nabízí velice zajímavý produkt, přinášející další zajímavé funkce, které by mohly být inspirativní do dalšího vývoje.

Firma Ki-Wi [3] nabízí kompletní řešení informačních panelů, jak autonomních, tak uživatelsky ovládatelných. Jejich nabídka je rozdělena do několika produktů podle funkcí, čemuž může napovídat fakt, že využívají více aplikací, které by aplikace mohla obsahovat dohromady.

Seznam produktů:

- Ki-Wi Server
- Ki-Wi ActiveCatalog
- Ki-Wi MiniPlayer
- Ki-Wi Kiosk
- Ki-Wi NaviSystem
- Ki-Wi Player
- Ki-Wi Analytics
- Ki-Wi EyeMetrics

Firma své produkty míří rovnou do prodejních míst, obchodních center ale také do úředních prostor, městům k digitalizaci úřední desky a zprávám z vedení města. Ke každému použití odpovídá jiný produkt. Pro klasické autonomní systémy se využívá produktu MiniPlayer nebo Player, produkt MiniPlayer je vyvinut na platformě Android společnosti Google, u produktu Player je jen informace o speciálně upraveném operačním systému. Kiosk je již interaktivní aplikace, stavějící své použití na internetovém prohlížeči, nejspíše omezeném pro skupinu serverů, na které může uživatel přistupovat.

Ki-Wi Server je aplikace sloužící pro správu produktů, změnu obsahu z jednoho místa. Tímto produktem propojují všechny ostatní a je zaměřen čistě jen na administraci.

Zajímavým produktem je NaviSystem, který zlepšuje orientaci v budově, umí vyhledávat místnosti a dát uživateli radu, kterým směrem se vydat. Produkt ActiveCatalog, kombinuje prvky internetového obchodu do kamenného, s možnosti hledání produktů a zobrazení informací.

Dalším zajímavým produktem je Analytics, který umí sledovat data z jednotlivých produktů a vytvářet statistiky. Data pro analýzu využívaní informačního panelu se dají získávat jen s aktivním zapojením uživatele, u autonomních panelů není dostatek parametrů, které lze jednoduše sledovat. Tento problém ovšem velmi zajímavě řeší produkt EyeMetrics, který využívá kameru nebo zařízení Kinect (herní ovladač od Microsoftu,

[3] www.ki-wi.cz

využívající snímaní dění k interpretací akcí, hráč tedy využívá svého pohybu) k vyhodnocení situace před informačním panelem a možnosti cílení obsahu informačnímu panelu pro pohlaví, věkovou kategorii, kterou dokáže rozlišit. Zajímavá se jeví detekce pozornosti a očního kontaktu, podle které se již můžou optimalizovat části informačního panelu.

Bez přesné technické dokumentace se dá jen velice složitě odhadovat, o jak kvalitní produkty jde a jaké jsou jejich schopnosti. Bohužel v internetové prezentaci není o těchto informacích nic napsáno. Závěrem je tedy to, že firma nejspíše disponuje velice podobným produktem, je zřejmá znalost trhu, dána rozsáhlými referencemi, pro který své produkty připravuje a tím pádem některé funkce jsou pokročilejší. Velice přínosný může být také fakt, že nabízí pro své produkty již hotové zařízení, jak pro interní, tak pro externí použití.

6 Implementovaná část

6.1 Použité technologie

Aplikace je vyvinuta v popsaných technologiích v teoretické části aplikace. Při začátku mé práce, Visual Studio mé používané verze, umožňovalo vytvořit projekt pouze pro verzi 2.0 WinJS.

HTML5 a CSS3 je využito pro celé grafické prostředí. Obecně jsou jejich možnosti pro tvorbu vzhledu prostředí je velmi pokročilé a velmi dostatečné pro zamýšlenou funkčnost aplikace. Kombinací těchto technologií a principů, jakým způsobem jsou vyvíjeny dnešní moderní internetové stránky, je možné vytvořit jakékoliv grafické efekty. Využití těchto technologií je také nepředstavitelně časově náročné oproti vývoji vlastních vykreslovacích systémů, s podmínkou tak vysoké ohebnosti všech grafických parametrů.

K JavaSriptovým knihovnám byly přidány další, pro ulehčení práce s DOM, známá knihovna jQuery[4]. Tato knihovna je velice používána pro práci v klasických webových aplikacích a je oblíbena přes pro svou malou velikost. Hlavní výhodou je usnadnění práce se selektory oproti klasickému JavaScriptu, je tvořena objektem *jQuery*, který je také přístupný přes alias *$*. jQuery také odděluje "chování"od HTML struktury, podobně jako CSS odděluje "zobrazovací"charakteristiky.

Další využitou JavaScriptovou knihovnou je DateJS[5], ulehčující práci s daty, jejich formátováním a dalšími užitečnými operacemi. Pro otevírání ZIP balíčků, jsem vybral knihovnu s názvem JSZip[6], který umí nejenom otevírat, ale zároveň i vytvářet.

Poslední využitou knihovnou je Less.js[7], pro překlad LESS v klientské částí aplikace, zamýšlenou pro kompilaci LESS až v uživatelově prohlížeči.

6.2 Vstupní soubor

Jak již bylo řečeno, pro definici vzhledu a chování informačního panelu je nutno aplikaci předat konkrétní vstupní soubor, který může být doplněn i o další soubory, jako jsou obrázky, zvuk nebo video. A proto jsem se rozhodl pro použití balíčku. Ideálním způsobem se jevilo využití i nějakého komprimačního algoritmu a proto byl vybrán velmi rozšířený formát ZIP.

Využitím archívu se také vyřešil problém s bezpečnostním omezením aplikace. Uživatel vybere jen jeden soubor, který je aplikací zpracován, v opačném případě by aplikace musela uživatele vybídnout k vyhledání všech potřebných souborů ručně. Další výhodou se může také jevit při získávání aplikací z jiného než lokálního zařízení.

Balíček může mít libovolnou strukturu, ale je nutno dodržet základní požadavek a to, že soubor, ze kterého bude vstup čten musí být uložen v kořenu a musí se jmenovat *panel.xml*.

[4] http://www.jquery.com
[5] http://www.datejs.com
[6] https://stuk.github.io/jszip
[7] http://lesscss.org/#client-side-usage

6.2.1 Význam částí vstupního souboru

Vstupní soubor je limitován svou pevnou strukturou rozdělenou do několika částí. Tuto strukturu si obecně můžeme prohlédnout v následujícím seznamu. Tato pevná struktura by měla udržet soubor přehledný.

- Část celkového nastavení informačního panelu
  - Globální nastavení
  - Místo pro vlastní stylopis
- Část pro jednotlivé obrazovky informačního panelu
  - Schéma první obrazovky
    - Vzhled obrazovky
    - Chování
  - Schéma druhé obrazovky
    - Vzhled obrazovky
    - Chování
  - Další obrazovky
- Část pro celkové chování aplikace
  - Jednotlivé akce

Nejdůležitější součástí celého konfiguračního souboru je část pro jednotlivé obrazovky informačního panelu, v této části nalezneme popsány jednotlivé vzhledy stránek informačního panelu a jejich chování. Počet jednotlivých stránek není omezený.

Další, ale již nepovinnou části, je část pro celkové nastavení informačního panelu, v konfiguračním souboru je uvedena jako první jen z důvodu přehlednosti. Tato část je připravena k tomu, aby obsahovala veškeré nastavení pro celý informační panel, tudíž může ovlivnit každou komponentu. To znamená, že zde můžeme konfigurovat nastavení pro výchozí jazyk informačního panelu, formát data a času. Součástí této sekce je také místo pro vložení vlastního stylopisu v LESS.

Poslední částí je část pro ovlivnění chování celé aplikace. Je stejná jako chování každé obrazovky a může obsahovat stejné akce. Z obecného hlediska může nahradit jednotlivý popis chování akcí u jednotlivých obrazovek, ale hlavním záměrem je možnost změnit aktivní obrazovku, pokud je potřeba aby obrazovka měla krátké cyklické chování a záměrem je měnit obrazovky v časovém intervalu. Tímto způsobem se ulehčí složité přepočítávání trvání a zasazování čekání do lokálního chování, nebo určování, kolikrát se tato obrazovka má opakovat aby naplnila požadovaný čas.

6.2.2 Proces zpracovávání vstupního souboru

Konfiguračním souborem je tedy ZIP balíček, který po výběru v aplikaci se kopíruje do temp složky WinJS aplikace. V této složce je balíček rozbalen a obsah překopírovaný do lokální složky. Zde již je aplikací otevíraný soubor *panel.xml*.

V této chvílí aplikace začíná zpracovávat tento soubor. Zpracování probíhá po logických částech popsaných v předešlé kapitole. Protože aplikace využívá HTML stránky k zobrazení, část s LESS stylopisem se kompiluje do CSS a je vložena přímo do stylovacího tagu v zobrazení.

Ostatní prvky je již nutno procházet a podle nich vytvářet adekvátní objekty, představující jejich funkčnost. Nejprve se zpracovávají vzhledové části informačního panelu, postupně podle stránek. Funkční objekty jsou ve stránce označeny vlastním tagem, tyto objekty mají pevně definovaný HTML kód, který se místo nich vloží do výsledné stránky. Ostatní tagy jsou klasické tagy jazyka HTML, a vkládají se přímo.

Zpracování objektu začíná čtením jeho atributů, *id* a *class* se zachovávají a elementu se vkládají přímo. Tímto způsobem si atribut *id* zachovává svou funkčnost, jako v klasickém HTML. Speciální vlastnosti jsou již závislé na dané konkrétní komponentě, obecně je, ale důležitý atribut value, představující hodnotu daného prvku. Může jít o text, jméno obrázku a podobně. Komponenta těchto atributů může mít více. Atributy nejsou však jediným způsobem předání informací komponentě, mohou být předány jako "child element"XML čehož je využito u komponenty galerie. Pokud tato hodnota se dále nevyužívá k dalšímu zpracování (například hodnotou je odkaz pro stažení obsahu), je tato hodnota vložena do HTML elementu a dále do stránky.

Identifikátor elementu je dále využit pro identifikaci při následné aktualizaci elementu. Tímto se dostáváme k části, kdy se zpracovává chování. Předpokladem je již zpracované rozložení, protože pro akce, kromě akcí tvořících čekání, jsou závislé na svém cíli. Podobně jako komponenty obsahují akce také své atributy definující cíl, novou hodnotu a typ akce. Tyto akce jsou buď obecné anebo speciální pro komponentu, kterou ovlivňují.

V aplikaci jsou implementovány také speciální akce ovlivňující tok zpracovávání. Inspirací jsou klasické podmínky a smyčky známé z programovacích jazyků. O těchto prvcích již bylo řečeno v návrhu aplikace a obsahují již předpřipravené typické podmínky závislé na čase. Detailnější popis je uveden v samostatné kapitole *Chování 6.2.3.2*.

Zpracováváním akcí se vytváří scénář, který je předán kontroléru, starající se o průběh chování pro každou stránku. Každé chování stránky je přiřazeno kontroléru, starající se o chování celého informačního panelu. V současné chvíli není možné zpracovávat více akcí najednou, proto se paralelní operace se musí zpracovávat sekvenčně.

6.2.3 Struktura aplikace

6.2.3.1 Komponenty Každá komponenta je potomkem třídy *Component*, která obsahuje vlastnosti a obecné metody, pro vytvoření a aktualizaci elementu. Z vlastností obsahuje zmíněný identifikátor a informaci o tom jaký HTML tag představuje. Kromě funkcí starajících se o zapouzdření objektu, obsahuje důležitou funkci *update*, která je volána s

polem atributů, ve kterém je definována akce změny. Funkce *update* se tedy stará o dynamičnost a je přepisovaná u specializovanějších komponent.

Metoda *toDefault* navrací komponentu do výchozího stavu, je volána pro každou komponenty aktuální stránky při zahájení zobrazování stránky znovu, aby se zachovala stejná opakovatelnost komponenty pro další zobrazení.

Pokud v konfiguračním souboru není u komponenty definovaný identifikátor je tento identifikátor vygenerován metodou *generateRandomId*.

Na této třídě jsou založené komponenty *TextBox*, *ImageBox*, ale také třída *Page*. Návrh těchto tříd je založený na návrhovém vzoru kompozit. Třída *Page* je nositelem dceřiných komponent, je prozatím jediná s touto vlastností.

Doposud byla řeč pouze o komponentách, které mění svůj obsah pouze na základě akcí definovaných v konfiguračním souboru. Těmito akcemi jsme schopni zařídit i komponentu, která mění svůj obsah v časových intervalech, ale tohle řešení je značně nepohodlné. Za tímto účelem je implementována třída *UpdatingComponent*, která má volání funkce *update* cyklicky v daných časových intervalech. Tento čas je plně ovlivnitelný zadáváním v rozmezí milisekund až hodin. Příkladem využívajícím tuto implementaci může být galerie nebo komponenta zobrazující aktuální čas.

Posledním typem komponent jsou již jen komponenty závislé na obsahu mimo aplikaci, jak již bylo v návrhu popsáno. Jde o komponentu závislou na API určité služby. Tyto komponenty jsou také potomky třídy *UpdatingComponent*, ale jejich funkčnost je oddělena do dalších pomocných tříd, aby byl dodržen předpoklad, aby se každá třída starala jen o jeden úkol. Pro tento úkol byla navržena třída *AjaxHelper*, která se stará o požádání serveru o data. Dceřiné třídy již obsahují algoritmy zpracovávající odpověď. Protože tyto API musí být implementovány podle dokumentace a stěží se dají vymyslet abstraktně, musí být navrženy přesně na míru. Příkladem pro může být počasí, v aplikaci je implementován pomocník pro službu OpenWeatherMap [8] Struktura odpovědi může být jiná pro každou službu, ale přijaté data jsou typově stejné, proto výsledné informace se již ukládají do připraveného objektu, který je předávaný do adekvátní komponenty. Komponenty pro počasí jsou *CurrentWeatherWidget* pro aktuální počasí a *WeatherForecastWidget* pro předpověď.

Avšak existuje také služba, která má svou odpověď standardizovanou. Příkladem této služby je RSS. Zde již je možná implementace starající se o jednu pomocnou třídu přijímající RSS, třídy držící jeden záznam a komponenty zobrazující tento obsah. Aplikace obsahuje implementovanou verzi 2.0 [9]

Aplikace obsahuje následující komponenty:

- Jednoduché komponenty
  - Page
  - TextBox
  - ImageBox

[8] www.openweathermap.com je služba nabízející data o počasí zdarma, tak i v placené variantě. Nabízí data pro celý svět, aktuální počasí tak i předpověď až na 16 dní.

[9] Detail specifikace RSS 2.0 http://validator.w3.org/feed/docs/rss2.html

  - VideoElement
  - AudioElement

- Komponenty samočinně se aktualizující
  - RollerText
  - ImageGaleryBox
  - DateTimeBox
  - DateClockWidget

- Speciální dynamické komponenty
  - RssComponent
  - CurrentWeatherComponent
  - WeatherForecastComponent

Posledními nezmíněnými jsou komponenty obalující video a audio prvky. Tyto komponenty umí přehrát prozatím jen lokálně uložené média v klasickém HTML5 přehrávači.

6.2.3.2 Chování Chováním v aplikaci je myšlen tok akcí, ovlivňující komponenty informačního panelu, které jsou drženy v objektech třídy *BehaviourController*. Třída se stará o zpracování akcí z konfiguračního souboru a vytvoření adekvátních objektů reprezentující jednotlivé akce. Tato třída má za úkol také spouštět jednotlivé kroky scénáře a po dokončení aktuálního, přejít do navazujícího, anebo začít celý scénář opakovat. Z tohoto důvodu třída obsahuje referenci na objekt stejného typu, nazvanou *afterDoneReturn*, referující na další kontrolér.

Třída pro držení toku akcí obsahuje pole objektů *AbstractAction*, představující rodičovskou třídu pro všechny akce. Tato třída abstrahuje veškeré akce pro aktualizaci daných objektů, ale nestará se samotnou aktualizaci dat, pouze předává nové data komponentě. V jedné chvíli může být aktivní pouze jediná akce daného kontroléru. Samotná akce má referenci na svůj kontrolér, po dokončení své akce říká kontroléru, aby vykonal další akci.

Obyčejnou akcí, která nemá svůj cíl, tudíž neaktualizuje žádnou komponentu, je akce čekání. Objekt realizující tuto akci je instancí třídy *WaitAction*. Zpoždění touto akcí je realizováno funkcí *setTimeout*, která po uplynutí intervalu říká kontroléru, že může přejít na další akci. Při použití této funkce však nastává problém, pokud aplikace již přejde do další fáze, např. přechod na další stranu. Z tohoto důvodu je důležité mít možnost interval přerušit, proto kromě funkce *doAction* realizující zmíněné vykonání, obsahuje také funkci *disableActions*, která je určena ke stornování průběhu. Získaní času od uživatele pro čekání je vymyšleno dostatečně intuitivně, umožňuje zadávání od milisekund po hodiny i s kombinací jednotlivých hodnot.

Další akce už obsahují referenci na danou komponentu, které volají s novými parametry funkci *update*. Touto vlastností jsou sjednocené aktualizace komponent stejně i pro prvotní inicializaci.

Pro jednoduché ovlivnění toku scénáře je zde implementován cyklus *ForLoop* s daným počtem opakování.

Pro složitější ovlivnění toku jsou implementovány akce *IfCondition* a *WhileLoop*. Obě akce přijímají parametry představující podmínky. Podmínky mohou představovat jakoukoliv akci, která je možná realizovat pomocí JavaScriptu, ale je nutné tyto podmínky mít realizované již v aplikaci. Obecné rozšiřování těchto podmínek je zamýšleno velice jednoduše, vytvořením třídy rozšiřující *Condition* s metodami, kde jméno metody představuje také jméno podmínky pro informační panel.

Vyhodnocování podmínek obstarává objekt *ConditionResolver*, kterému jsou předány a ten volá jejich příslušné funkce. Pro vyhodnocení podmínek jako pravda, musí být všechny splněny. Ze zvoleného zápisu podmínek jako atributy elementu, nelze tyto atributy opakovat a z tohoto důvodu nelze realizovat logické operátory. Obecně toto řešení není omezujícím, protože není záměrem tyto podmínky zbytečně komplikovat a pokud by nastala situace vyžadující složitější akci, může být implementována jako samostatná metoda v rozšiřující třídě.

Zvolením cesty s cílem co nejjednoduššího zadávání těchto podmínek, realizovanými metodami, je vhodné také již nějaké obsáhnout v aplikaci. Předpokládaným parametrem, který bude nejčastěji ovlivňovat zobrazování, podle mého názoru je časová doba. Informační panel může zobrazovat jiný obsah ráno a jiný zase odpoledne. Připraveny jsou tedy podmínky rozlišující denní dobu ve zvoleném hodinovém rozmezí, tak i dané pro přímou kontrolu jestli je ráno, večer a podobně. V podobném způsobu je zde varianta pro dny.

6.2.4 Konkrétní implementace zajímavé části

Při vývoji jsem se snažil o co nejjednodušší rozšiřitelnost aplikace, při implementaci připomínkovacího systému zmíněného v předešlé kapitole jsem narazil na problém, jak řešit přidávání nových funkcí. Chtěl jsem, aby podmínky mohly být v samostatných třídách, podle kategorie kam spadají a zároveň aby se jednotlivé akce nemusely nikde registrovat.

Řešením je projít adekvátní třídy, její funkce a po nálezu ji zavolat. V obecném programování se tomuto principu říká reflexe, ale JavaScript není typickým představitelem toho, jak to v jiných jazycích funguje a má odlišný způsob tvoření instancí založených na prototypech, které jsou ve skutečnosti také objekty.

Ve funkci *getTypeCondition* jejíž implementaci si můžete prohlédnout v následující ukázce, je procházen objekt *window*, který drží veškeré objekty a hledá se objekt se jménem obsahujícím slovo "Condition". Toto slovo je tedy podmínkou pro to, aby metody této třídy byly brány jako podmínky. Následuje již podmínka, která kontroluje, zda je to hledaná funkce a vrátí její instanci do objektu *ConditionResolver*.

V případě, že metoda není nalezena, vrací se objekt typu *BaseCondition*, představující základní podmínku, která kontroluje jen, zda jméno funkce je stejné jako hodnota parametru.

```
private getTypeCondition(functionName: string, value: string ): Condition {
        for (var object in window) {
            var objectName: string = object.toString() ;
            if (objectName.indexOf("Condition") !== -1) {
                if (typeof window[objectName].hasMethod != "undefined" && window[objectName].
                    hasMethod(functionName)) {
                    return new window[objectName](functionName, value);
                }
            }
        }
        return new BaseCondition(functionName, value);
    }
```

Výpis 9: Vyhledávání funkce podle jména v objektech

6.3 Ukázkové použití

V rámci představení aplikace jsem připravil ukázkové použití informačního panelu pro nasazení do restaurace. Tento panel se skládá celkem z 2 stránek a na každé je použitý stejný základ, skládající se ze jména restaurace a aktuálního času. Na spodní levé straně je informace o ceně denního menu s animací běžícího textu. První stránka (obrázek 10) je tvořena galerií fotek restaurace a otevírací dobou, zároveň je zde informace o tom, zda je možné si dát denní menu z nabídky, které je dostupné v čase 10 až 13 hodin. Na další stránce (obrázek 11)je již výběr z nabízených jídel s fotografiemi těchto jídel. První stránka má limit 30 sekund, poté se přepíná na stránku, která je zobrazena minutu. Samotná restaurace společně s nabídkou je smyšlená.

Druhou ukázkou bych rád demonstroval použití v muzeu automobilů, konkrétně pro Kopřivnickou společnost Tatra. Vybral jsem si automobil Tatra 603 a připravil třístránkový informační panel. První stránka (obrázek 12) je věnovaná několika fotografiím a obecnými informacemi dostupných z wikipedie. Na další stránku jsem si dovolil použít dobovou reklamu natočenou společností Krátký film Praha, a.s.[10], pro reprezentaci funkčnosti video přehrávače.

Následující stránku (obrázek 13) tvoří další fotografie vozu s galerií. Na poslední stránku (obrázek 14) jsou umístěny technické informace o vozu.

6.4 Kompatibilita s ostatními zařízeními

Vyvinutá aplikace je kompatibilní se všemi dostupnými zařízeními, s operačním systémem Windows 8 a výše. Podporovány jsou i mobilní telefony, které však pouze kvůli velikosti displayů nejsou vhodnými adepty pro použití. Nejlepších výsledků pro kvalitní zobrazení se dostane na velkých úhlopříčkách.

Zajímavým zařízením se může jevit nejnovější verze mikropočítače Raspberry PI 2 Model B slibující podporu Windows 10. Použitím tohoto zařízení jsme schopni minimalizovat místo k umístění zařízení a přitom nabízí dostatečné periferie pro připojení všech

[10] http://www.kratkyfilm.cz/

Obrázek 10: První stránka panelu Restaurace

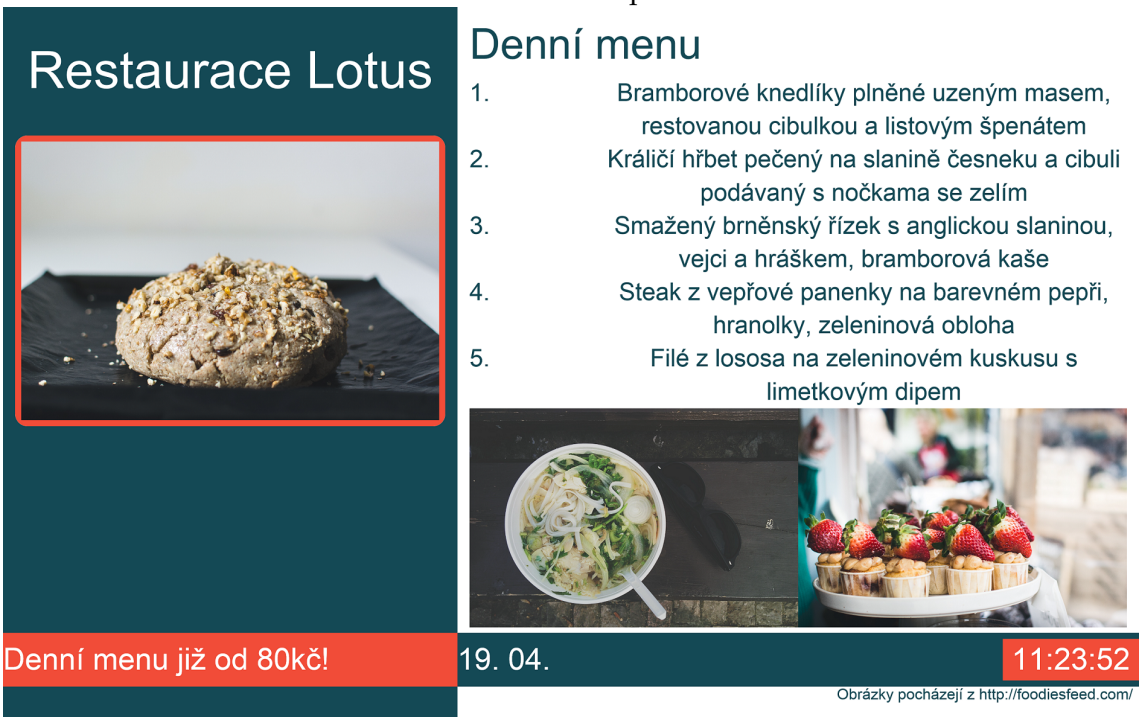
důležitých komponent. Samozřejmostí je také přístup k internetu. Nejjednoznačnější výhodou je fakt, že Windows 10 bude pro tyto zařízení dostupný zdarma, v tomto případě se cena zařízení bez zobrazovací jednotky dostává na £27 (asi 720Kč)[11].

Operační systém Windows však nemusí být omezující platformou pro použití, aplikace se pomocí několika úprav a přidáním WinJS knihovny ve verzi alespoň 3.0, dá exportovat pro použití v klasickém prohlížeči. Tomuto předpokladu se aplikace stává spustitelnou na jakémkoliv zařízení s podporou JavaScriptu v podporovaném prohlížeči. Podpora závisí na použitých knihovnách, například knihovna WinJS slibuje podporu ve všech hlavních prohlížečích napříč všem platformám[12].

[11] Zdroj: http://uk.rs-online.com/web/p/processor-microcontroller-development-kits/8326274/ (1.4.2015)

[12] Zdroj: https://github.com/winjs/winjs/wiki/Browser-Support (poslední editace článku 17. říjen 2014)

Obrázek 11: Druhá stránka panelu Restaurace



Obrázek 12: První stránka panelu Tatra 603

Obrázek 13: Druhá stránka panelu Tatra 603

Obrázek 14: Poslední stránka panelu Tatra 603

7 Závěr

V úvodu práce jsem se zaměřoval na platformu jako celek. Microsoft se snaží konkurovat ryze mobilním systémům svou politikou vlastnit operační systém jednotný pro veškeré zařízení. Z tohoto důvodu vzniklo také nové prostředí ModernUI, které přišlo se systémem Windows 8. V rámci tohoto systému obsahuje také velice kvalitní software pro vývoj, kterým je Visual Studio, snažící se co nejvíce zjednodušit práci. Při vývoji se s ním dobře pracovalo a využil jsem také dostatečně jeho ladících nástrojů, které jsou dobře připraveny i pro HTML5 založené aplikace.

V rámci práce jsem využil knihovnu WinJS ve verzi 2.0, který pracuje se standardem HTML5. Zdál se velice vhodným pro navrhované použití, protože dovoluje implementovat tuto aplikaci jako speciálním typ webové aplikace. Použitím tohoto řešení se zjednodušila práce nejenom při návrhu aplikace, ale bude velice přínosnou pro tvorbu specifických informačních panelů do budoucna, protože tuto práci bude moci obstarat člověk, který se stará o kódování webových aplikací. Výhodou je také, že není potřeba znovu, doslova řečeno, vynalézat kolo a vymýšlet použitelný způsob pro vykreslování grafického prostředí. Nejenom v tomhle ohledu jsem se snažil co nejvíce využít technologií, které jsou již zaběhnuté, a tímto faktem je při vytváření informačního panelu, nutná pouhá znalostí principů HTML, CSS (volitelně LESS) a XML.

K samotnému návrhu aplikace si troufám tvrdit, že byl vytvořen dobře, mým názorem je, že svou konstrukcí je použitelný pro následující rozšiřování veškerých částí. Aplikace implementuje většinu navrhovaných komponent, pro základní použití, jak bylo představeno v kapitole s ukázkami, které mohou být dostatečně vypovídající o možném použití. Obecně však obrázky nejsou dostatečně vypovídající. Budoucí možný postup již byl zmíněn a další funkce mohou přibývat. Je však samozřejmostí, že aplikace není ve finálním stavu a mohou se vyskytnout chyby. Doufám, že tato situace však nenastane. Aplikace v současném stavu neřeší validaci vstupního souboru, která však nemusí znamenat omezení, je nutná pouze dostatečná pozornost při vytváření konkrétních informačních panelů.

V rámci návrhu jsem také prozkoumal potencionální konkurenci. Nalezeno bylo několik firem nabízejících podobně zaměřený produkt. Většina firem však nabízí negrafické informační panely vhodné spíše na nádraží s odjezdy vlaků atd. Firma s vlastním řešením podobné aplikace, nabízející také i vlastní zařízení, ve kterých je software spuštěný, je možnou konkurencí. O použitých technologiích se však nezmiňují a není jasné, jakých kvalit dosahují. Firma však nabízí tyto produkty i s vytvořením a následnou správou. Tento způsob je možný, ale cílem této práce byla aplikace umožňující návrh a vlastní konfigurovatelnost.

8 Reference

[1] Microsoft, *Readme.md winjs/winjs* [Online] 26.2.2015 [cit. 2015-02-28], Dostupné z: https://github.com/winjs/winjs

[2] Microsoft, *Try. WinJS.* [Online] 26.2.2015 [cit. 2015-02-28], Dostupné z: http://try.buildwinjs.com/

[3] The Register, *Microsoft: It was never 'Metro,' it was always 'Modern UI'* [online]. [cit. 2015-02-28], Dostupné z: http://www.theregister.co.uk/2012/08/10/metro_is_modern_ui_now

[4] Windows návody , *Co je složka AppData?* [online]. [cit. 2015-02-28], Dostupné z: http://windows.microsoft.com/cs-cz/windows-8/what-appdata-folder

[5] Zdrojak.cz, *K čemu je dobrý TypeScript* [online]. [cit. 2015-02-28], Dostupné z: http://www.zdrojak.cz/clanky/k-cemu-je-dobry-typescript/

[6] Tolerik Developer Network, *WinJS 3.0 Shows Industry Shift Towards JavaScript* [online]. [cit. 2015-02-28], Dostupné z: http://developer.telerik.com/featured/winjs-3-0-shows-industry-shift-towards-javascript/

[7] Less , *Getting started* [online]. [cit. 2015-02-28], Dostupné z: http://lesscss.org

[8] Less, *Použití na klientské straně* [online]. [cit. 2015-02-28], Dostupné z: http://www.lesscss.cz

[9] Designmodo , *Modern UI Style Design by Microsoft* [online]. [cit. 2015-02-28], Dostupné z: http://designmodo.com/modern-ui/

[10] wikipedia.cz , *API* [online]. [cit. 2015-02-28], Dostupné z: http://en.wikipedia.org/wiki/API

[11] wikipedia.cz, *Debugger* [online]. [cit. 2015-02-28], Dostupné z: http://cs.wikipedia.org/wiki/Debugger

[12] wikipedia.cz, *Microsoft Visual Studio* [online]. [cit. 2015-02-28], Dostupné z: http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio

[13] wikipedia.en, *Microsoft Visual Studio* [online]. [cit. 2015-02-28], Dostupné z: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[14] Developer network, *Začínáme s nástrojem Blend for Visual Studio 2013* [online]. [cit. 2015-02-28], Dostupné z: https://msdn.microsoft.com/cs-cz/library/jj171012.aspx

[15] wikipedia.com , *Open weather map* [online]. [cit. 2015-02-28], Dostupné z: http://en.wikipedia.org/wiki/Openweathermap

[16] wikipedia.cz , *jQuery* [online]. [cit. 2015-02-28], Dostupné z: http://cs.wikipedia.org/wiki/JQuery

[17] wikipedia.com , *jQuery* [online]. [cit. 2015-02-28], Dostupné z: http://en.wikipedia.org/wiki/JQuery

[18] The Verge , *Microsoft to support Raspberry Pi 2 with a free version of Windows 10* [online]. [cit. 2015-02-28], Dostupné z: http://www.theverge.com/2015/2/2/7962179/raspberry-pi-windows-10

[19] Blog Windows , *A Preview of WinJS 4.0* [online]. [cit. 2015-02-28], Dostupné z: https://blogs.windows.com/buildingapps/2015/03/27/a-preview-of-winjs-4-0/

A Dokumentace

A.0.1 Vytváření konfiguračního souboru

Konfigurační soubor je obsažený v archívu, který obsahuje také obrázky a další multimédia. Archív musí tedy nutně obsahovat soubor *panel.xml*. Tento soubor je nositelem veškeré funkčnosti, je rozdělen do několika funkčních celků. Jeho základní struktura je vyobrazena v následující části. Využívá XML formátu a je nutné dodržet přesný tvar.

```
<?xml version="1.0" encoding="utf−8" ?>
<information−panel>
  <settings>
    <style>
    </style>
  </settings>

  <pages>
    <page number="1">
      <layout>
      </layout>
      <behaviour>
      </behaviour>
    </page>
  </pages>

  <scenario>
  </scenario>
</information−panel>
```

Výpis 10: Základní struktura konfiguračního souboru

Rozložení a vzhled celého informačního panelu je zapisován kombinací HTML a CSS. Vlastní stylopis se vkládá do části *style* v části *settings*. K zápisu je možno využít jazyka LESS, který si aplikace sama přeloží.

Informační panel může obsahovat více stránek, které představují samostatné obrazovky. Umístění stránek je v tagu *pages*, pro každou stránku je nutný element *page* se svým číslem (parametr *number*). Číslo je zde pro identifikaci a také pro pozdější změnu stránky. Samotná stránka se skládá ze samotného vzhledu (element *layout*) zapsaný s použitím HTML a komponent a částí obsahující akce (*behaviour*). Komponenty jsou elementy se speciální funkčností a jsou ovládány akcemi a budou popsány v samostatné kapitole, ale již z tohoto zápisu je zřejmé že každá stránka je definována svým vzhledem a chováním.

Poslední částí je *scenario*, který obsahuje také akce. Akce v této části však neovlivňují samostatné komponenty, ale celou stránku. Typickou akcí je již řečena změna stránky, která je realizována tagem *change-page* s parametrem *to* a hodnotou, která představuje cílový identifikátor stránky. Aby změna nenastala přímo hned, je možno využít speciální akce realizující čekání pomocí tagu *wait*. Určení času čekání je realizováno použitím parametrů a je možné jej realizovat v řádech milisekund, sekund, minut a hodin. Následující

příklad bere předpoklad, že existují dvě stránky a vyobrazuje zápis čekání 30 sekund, změny na stránku číslo 2, další čekání trvající minutu a půl a změnu na první stránku.

```
<scenario>
   <wait seconds="30">
   <change-page to="2">
   <wait minutes="1" seconds="30">
   <change-page to="1">
</scenario>
```

Výpis 11: Použití akce změny stránky a čekání

V případě potřeby opakovat určité akce více krát za sebou, bez nutnosti duplikace kódu, je vhodné obalit akce blokem *for*. Počet opakování je dán parametrem *count*.

```
<scenario>
   <for count="5">
      <change-page to="2">
      <wait minutes="1" seconds="30">
   </for>
</scenario>
```

Výpis 12: Použití cyklu for

Další možností ovlivnění toku akcí je již s použitím podmínkového systému, tímto způsobem je možno tok rozdělit v závislosti na čase a datu. Následující přehled popisuje dostupné podmínky.

- *is-morning* - pravda, pokud je čas v rozmezí 4:00-8:00
- *is-forenoon* - pravda, pokud je čas v rozmezí 8:00-12:00
- *is-afternoon* - pravda, pokud je čas v rozmezí 12:00-16:00
- *is-early-evening* - pravda, pokud je čas v rozmezí 16:00-20:00
- *is-evening* - pravda, pokud je čas v rozmezí 20:00-0:00
- *is-night* - pravda, pokud je čas v rozmezí 0:00-4:00
- *in-hour-interval* - parametrem je předán čas, pro který podmínka má být pravdivá
- *today-is* - parametrem je anglické jméno dne, pro který podmínka má být pravdivá
- *month-is* - parametrem je anglické jméno měsíce, pro který podmínka má být pravdivá

Použití podmínek se dá rozdělit do dvou způsobů a to zda se mají akce opakovat, dokud zůstává pravdivou, nebo se má provést jen jednou. Akcí *while* se podmínka vyhodnocuje při každém projití začátkem a je opuštěna až poté co podmínka není pravdivá. Opakem je *if*, která je vyhodnocena jen jednou. Navíc může obsahovat událostí pro situaci kdy podmínka je pravdivá, tak také pro situaci, kdy ne. Nemusí obsahovat oba toky, stačí akce obalit elementem *true* nebo *false*.

```
<scenario>
    <while is−morning>
        <change−page to="1">
        <wait minutes="5">
        <change−page to="2">
        <wait minutes="5">
    </while>
    <while is−forenoon>
        <change−page to="3">
        <wait minutes="5">
        <change−page to="4">
        <wait minutes="5">
    </while>
</scenario>
```

Výpis 13: Použití cyklu while

```
<scenario>
    <while today−is="monday">
        <true>
            <change−page to="1">
            <wait minutes="5">
            <change−page to="2">
            <wait minutes="5">
        </true>
        <false>
            <change−page to="3">
            <wait minutes="5">
            <change−page to="4">
            <wait minutes="5">
        </false>
    </while>
</scenario>
```

Výpis 14: Použití if

A.1 Komponenty

Nyní bude následovat popis jednotlivých komponent, využitelných v současné verzi aplikace. Komponenta zastřešuje vlastní funkčnost a její použití by mělo být jasné z jejího pojmenování. Samotné komponenty jsou ovlivnitelné akcí *action*, která však nemusí mít stejné jméno. Tento způsob je zavedený pro přehlednost. Akce musí obsahovat parametr *target*, který odkazuje na ovlivňovanou komponentu. Některé komponenty však obsahují speciální akce.

A.1.1 TextBox

Tag: *panel-text*
Parametry: *value/text* - textový obsah
Použití: zobrazení textu
Akce: s parametrem *value/text* - nová hodnota

A.1.2 RollerTextBox

Tag: *panel-roller-text*
Parametry: *value/text* - textový obsah, *type* - orientace běhu (from-left-to-right/default)
Použití: zobrazení běžícího textu
Akce: možno využít parametry *value/text* nebo *type*

A.1.3 DateTimeTextBox

Tag: *panel-date-time-box*
Parametry: *format* - format zobrazeného času (možnost využití běžných notací)
Použití: zobrazení času a data, s možností vlastního formátování
Akce: možno využít parametry *format*

A.1.4 ImageBox

Tag: *panel-image*
Parametry: *value/image* - jméno obrázku
Použití: zobrazení obrázku
Akce: možno využít parametry *value/image*

A.1.5 ImageGaleryBox

Tag: *panel-image-galery*
Použití: zobrazení galerie obrázků, obrázky se vkládají jako cesta do samostatného elementu uvnitř
Akce: možno využít parametry *add/remove* pro manipulaci s obrázky

A.1.6 AudioElement

Tag: *panel-audio*
Parametry: *file* - audio soubor, *volume* - nastavení hlasitosti
Použití: komponenta pro přehrávání zvuku

Akce: nutno využít tagu *audio-action*, s parametry *file*, *volume*, *toTime* pro přesun na daný čas a *action* pro zastavení (pause), spuštění znovu (playAgain) a spuštění (start)

A.1.7 VideoElement

Tag: *panel-video*
Parametry: *file* - video soubor, *volume* - nastavení hlasitosti
Použití: komponenta pro zobrazení videa
Akce: nutno využít tagu *video-action*, s parametry *value/text*, *volume*, *toTime* pro přesun na daný čas a *action* pro zastavení (pause), spuštění znovu (playAgain) a spuštění (start)

A.1.8 RssComponent

Tag: *panel-rss*
Parametry: *source* - url adresa, *number-of-posts* - počet zobrazených článků, *display-images* zobrazení obrázků (true/false), *display-title* zobrazení titulků (true/false), *display-date* zobrazení data vydání, *display-text* zobrazení textu
Použití: komponenta získávající obsah z RSS zdroje, znovuzískaní dat je nastaveno na 5 minut
Akce: možno využít veškerých parametrů pro změnu nebo *refresh* pro vynucení aktualizace

A.1.9 CurrentWeatherWidget

Tag: *panel-weather-current*
Parametry: *language* - jazyk, *town* - město (např: Prague,cz), *coord* - souřadnice (oddělené středníkem)
Použití: komponenta pro aktuálního počasí
Akce: stejné jako parametry

A.1.10 WeatherForecastWidget

Tag: *panel-weather-forecast*
Parametry: *language* - jazyk, *town* - město (např: Prague,cz), *coord* - souřadnice (oddělené středníkem),*days* - počet dní (1-16)
Použití: komponenta pro zobrazení předpovědi počasí
Akce: stejné jako parametry

B CD-ROM

K bakalářské práci je přiložen CD-ROM obsahující aplikaci, veškeré zdrojové kódy a ukázkové informační panely.